VESC
Vedder Electronic Speed Control

WeGo Korea

ROS

WeGo

02

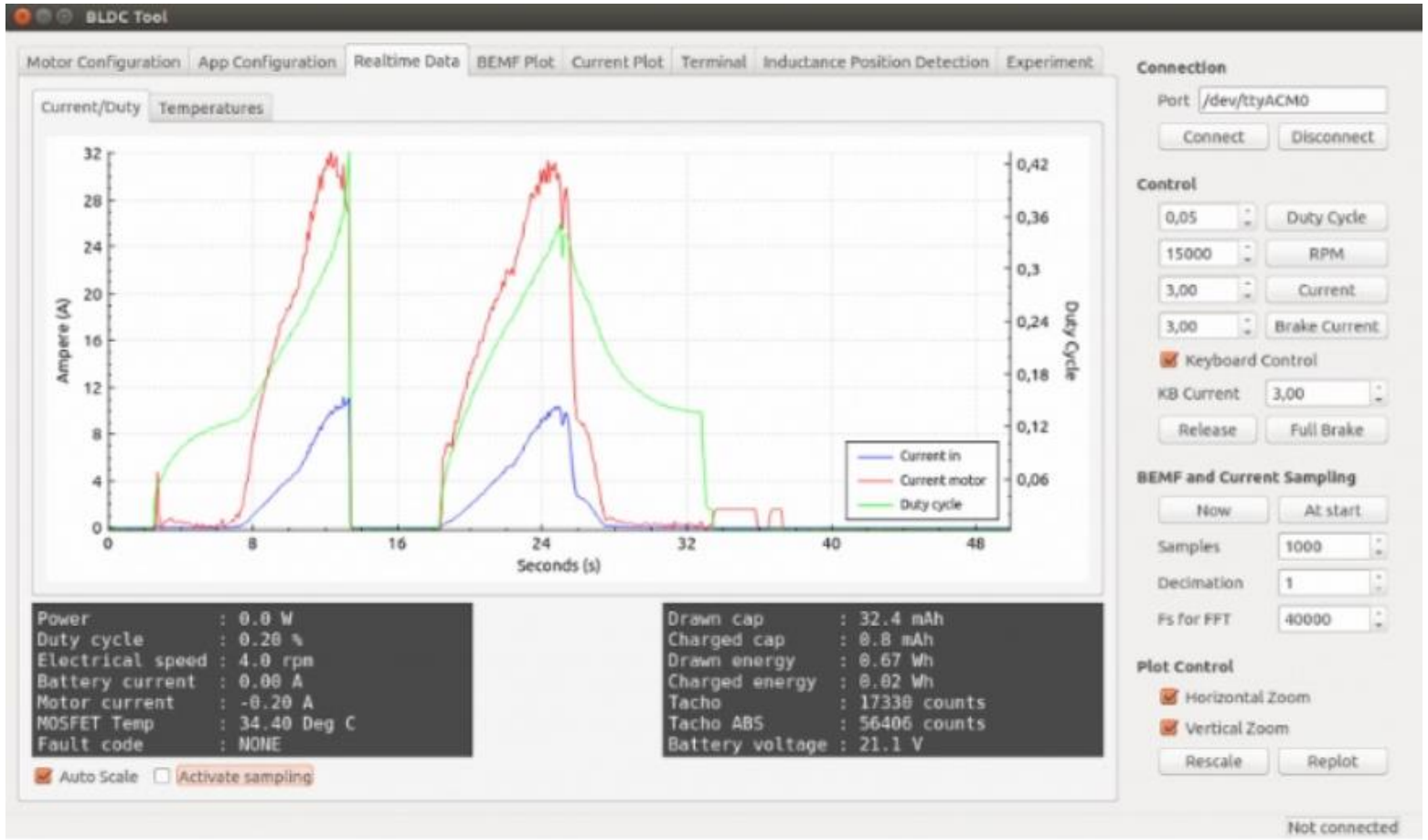
VESC using ROS

VESC_ubuntu16.04LTS

ubuntu 실행

VESC_ubuntu16.04LTS

- 펌웨어 변경 및 Duty 측정

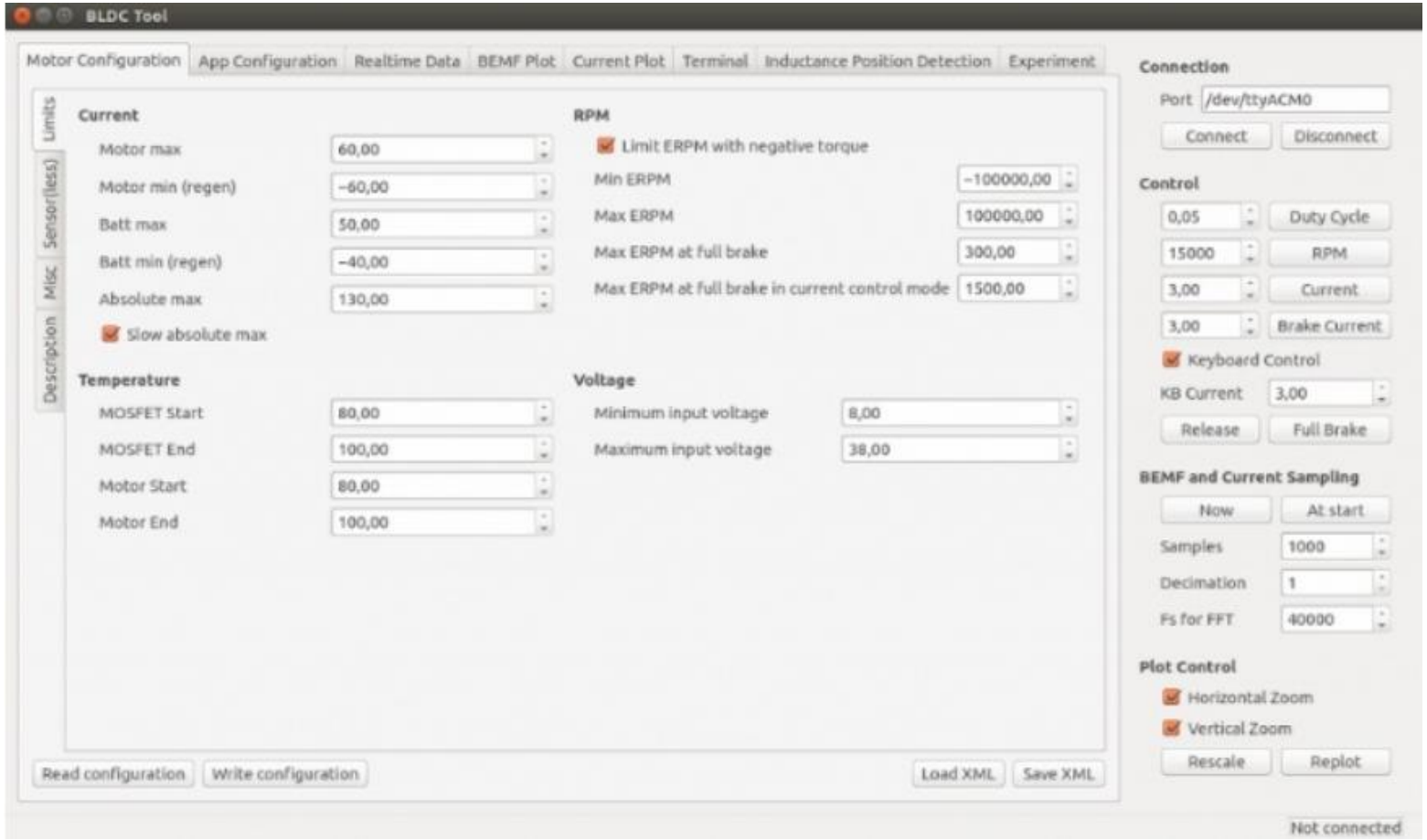


ROS

WeGo

VESC_ubuntu16.04LTS

- 속도 셋팅 및 펌웨어 업로드

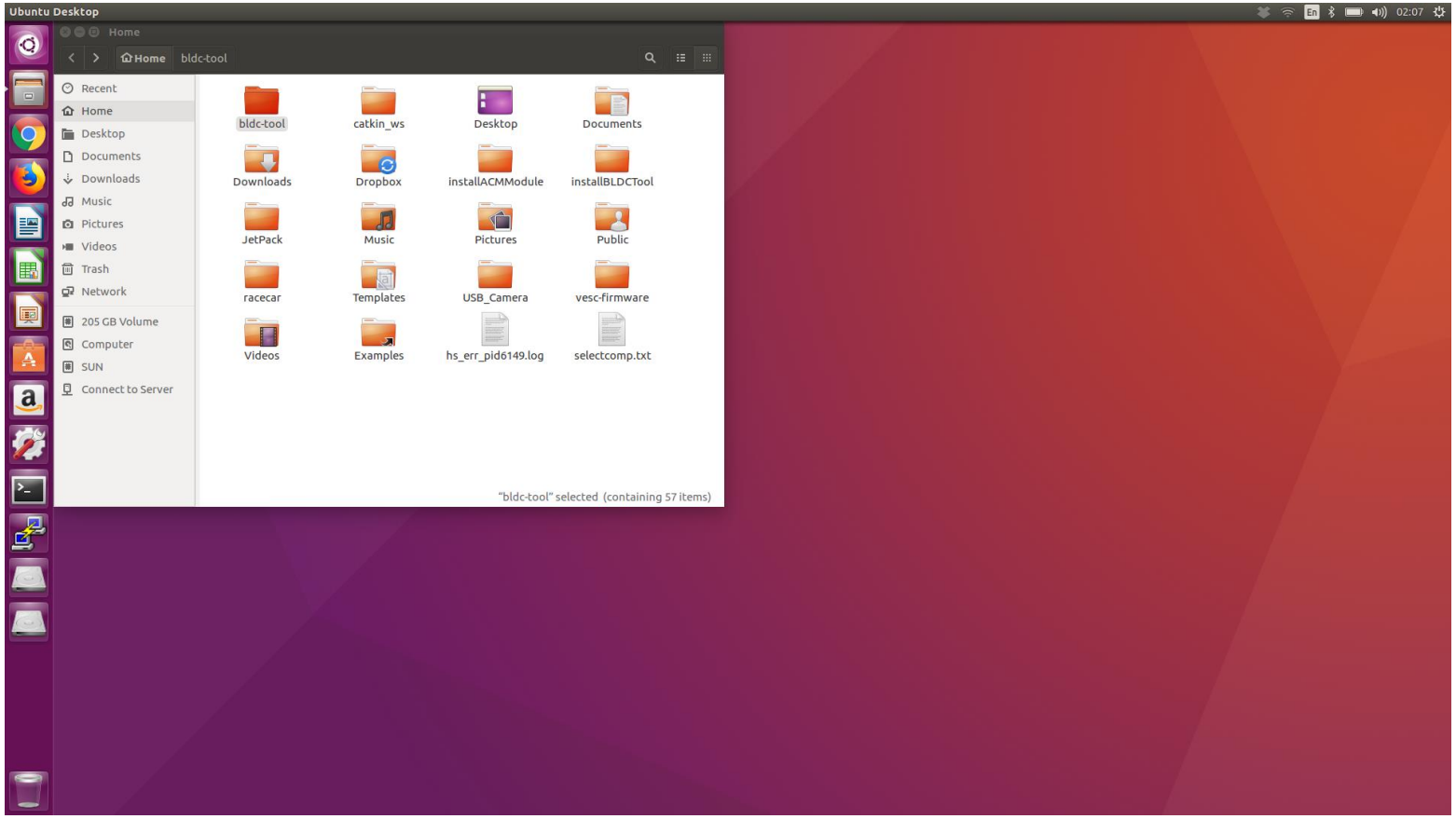


ROS

WeGo

VESC_ubuntu16.04LTS

- BLDC_Tool install

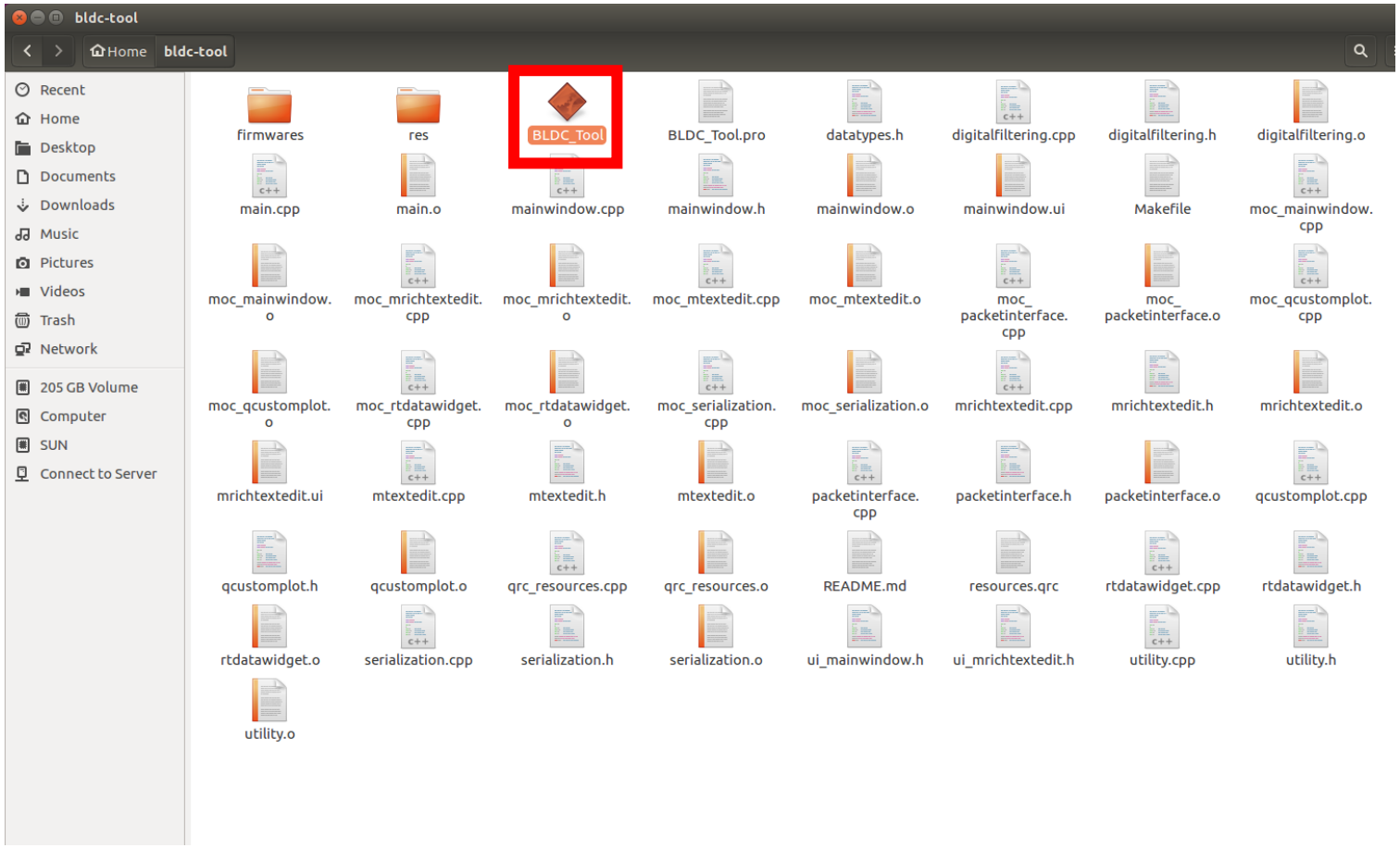


ROS

WeGo

VESC_ubuntu16.04LTS

- BLDC_Tool install

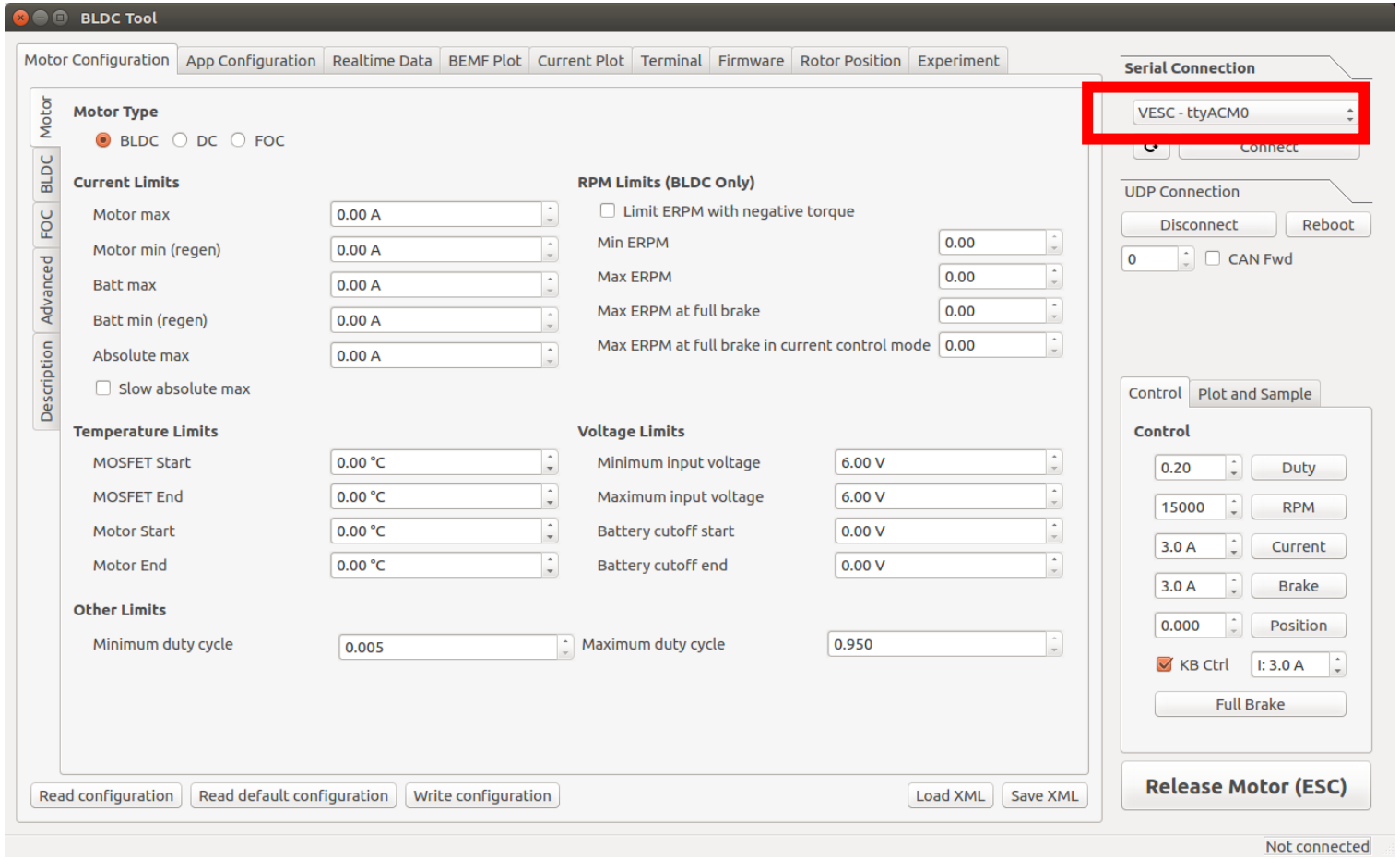


ROS

WeGo

VESC_ubuntu16.04LTS

- BLDC_Tool Connect

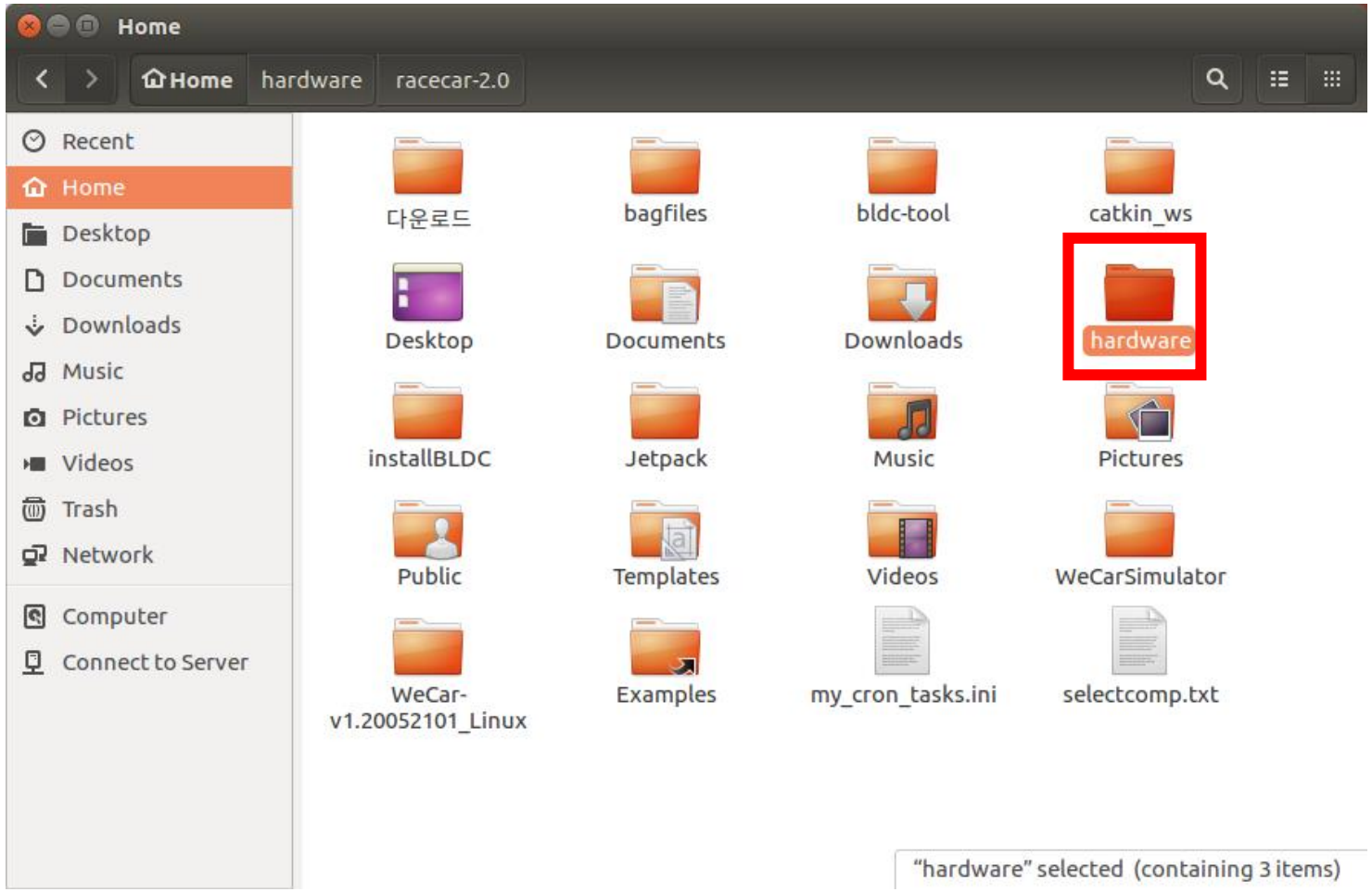


ROS

WeGo

VESC_ubuntu16.04LTS

- BLDC_Tool Connect_bin

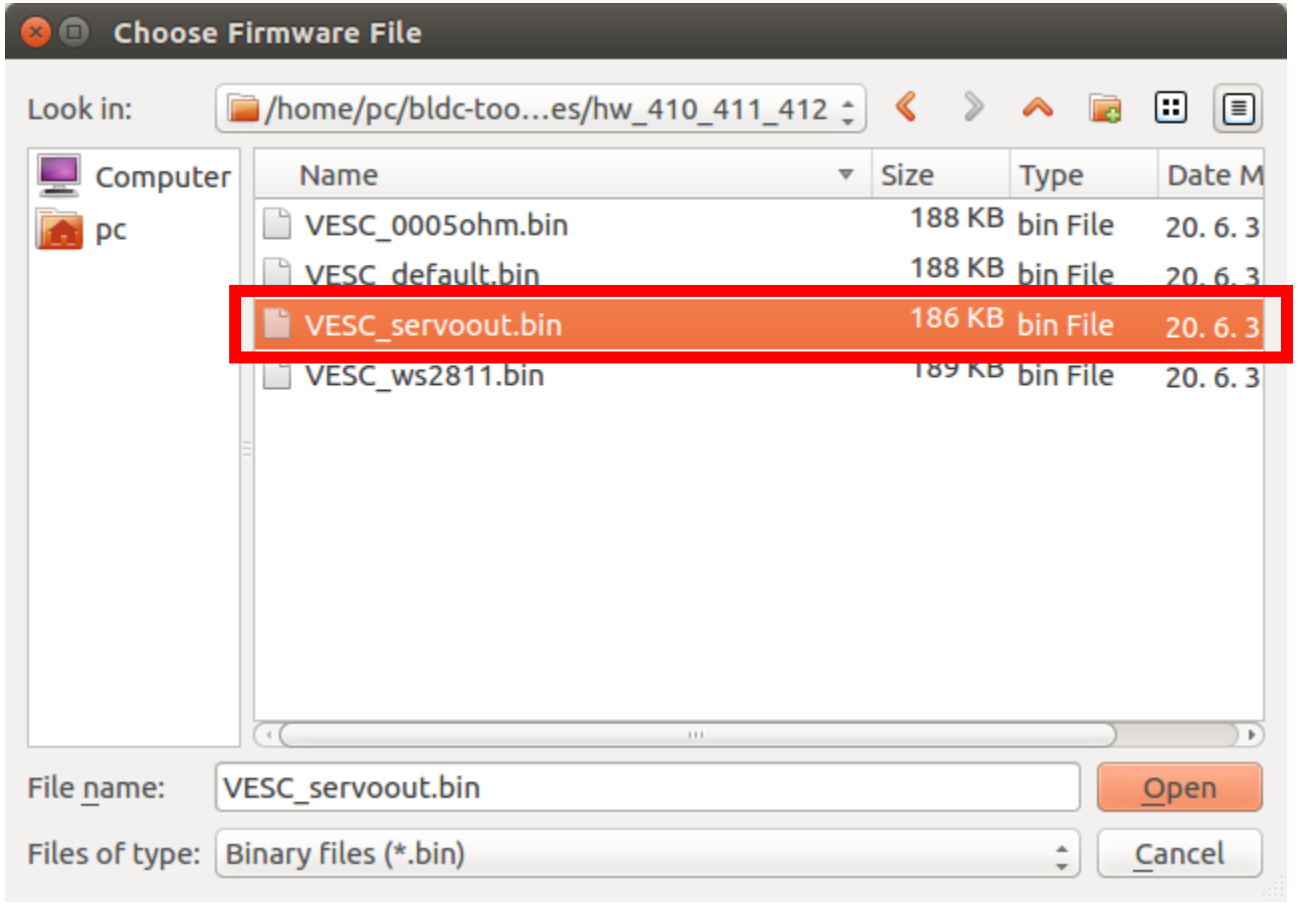


ROS

WeGo

VESC_ubuntu16.04LTS

- BLDC_Tool Connect_bin

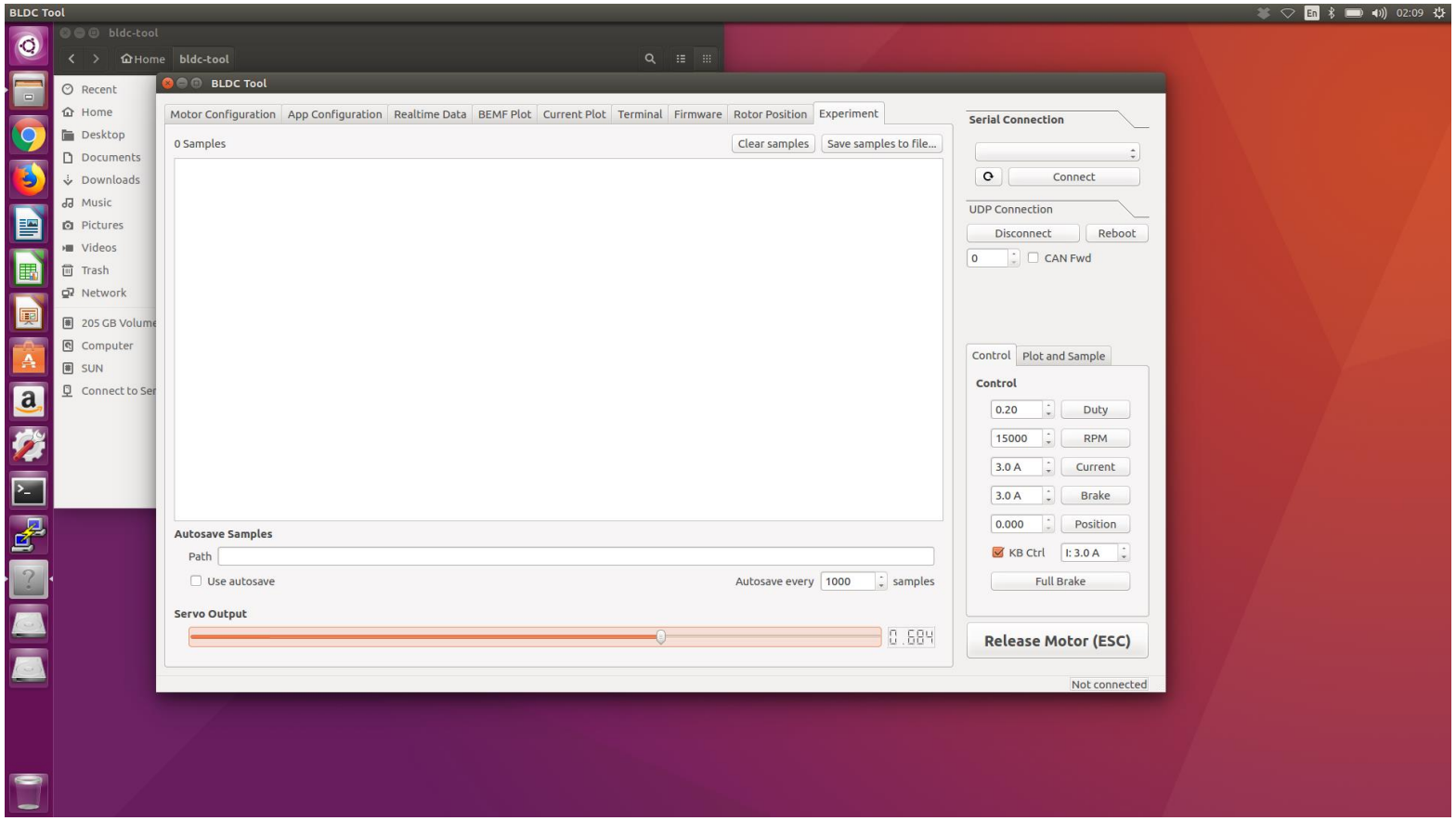


ROS

WeGo

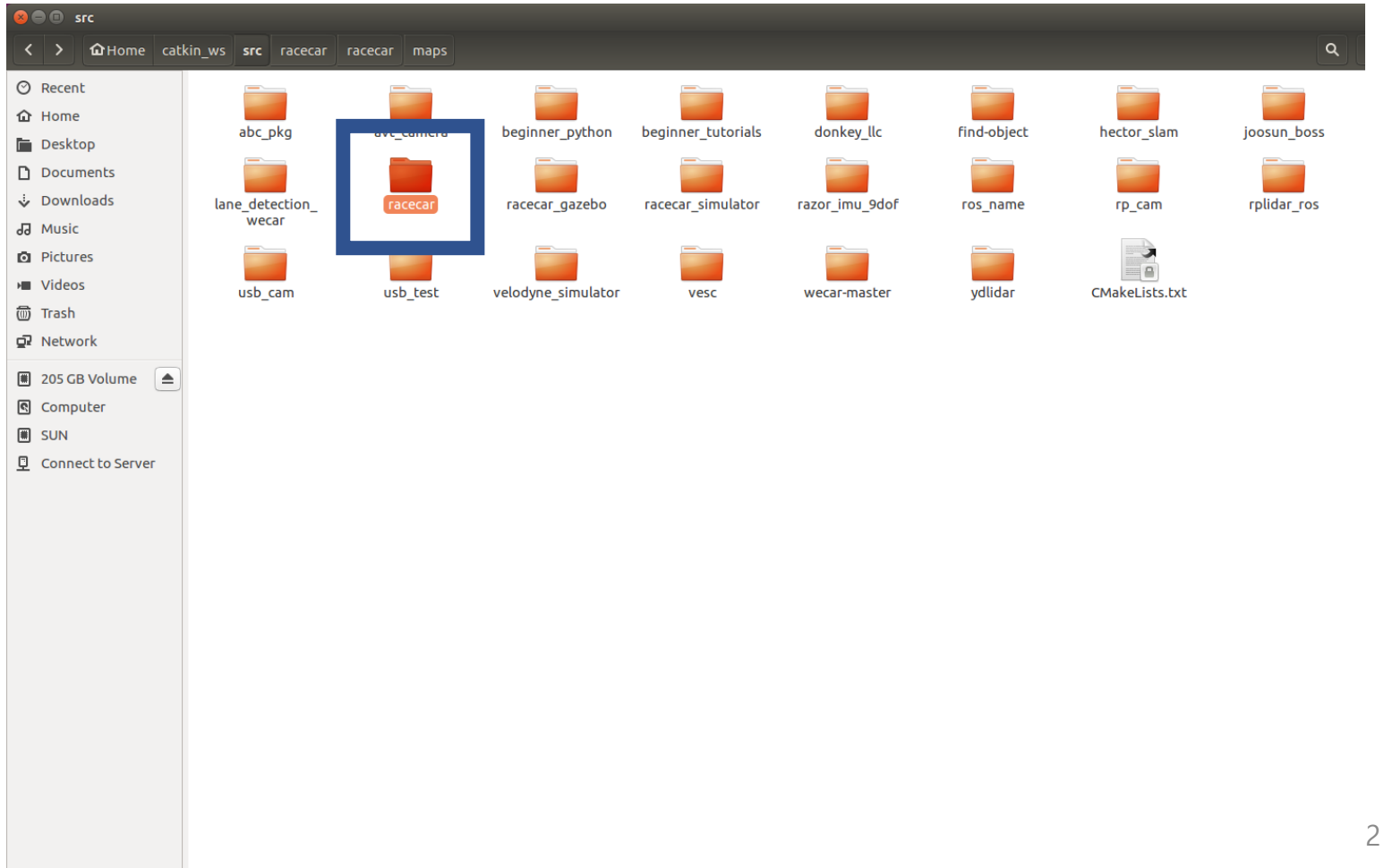
VESC_ubuntu16.04LTS

- BLDC_Tool Servo_motor



VESC_pkg

- racecar_pkg

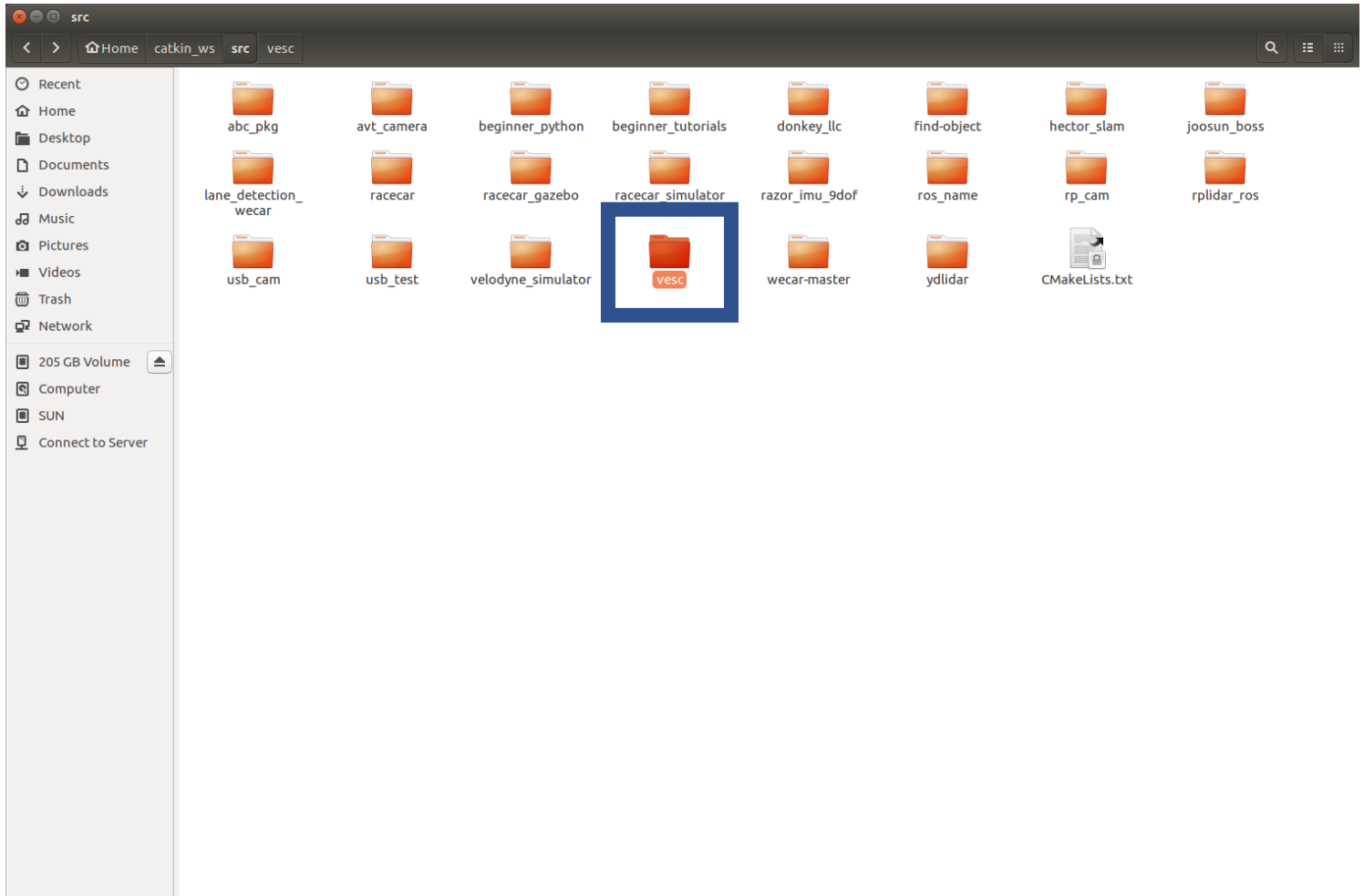


ROS

WeGo

VESC_pkg

- vesc_pkg

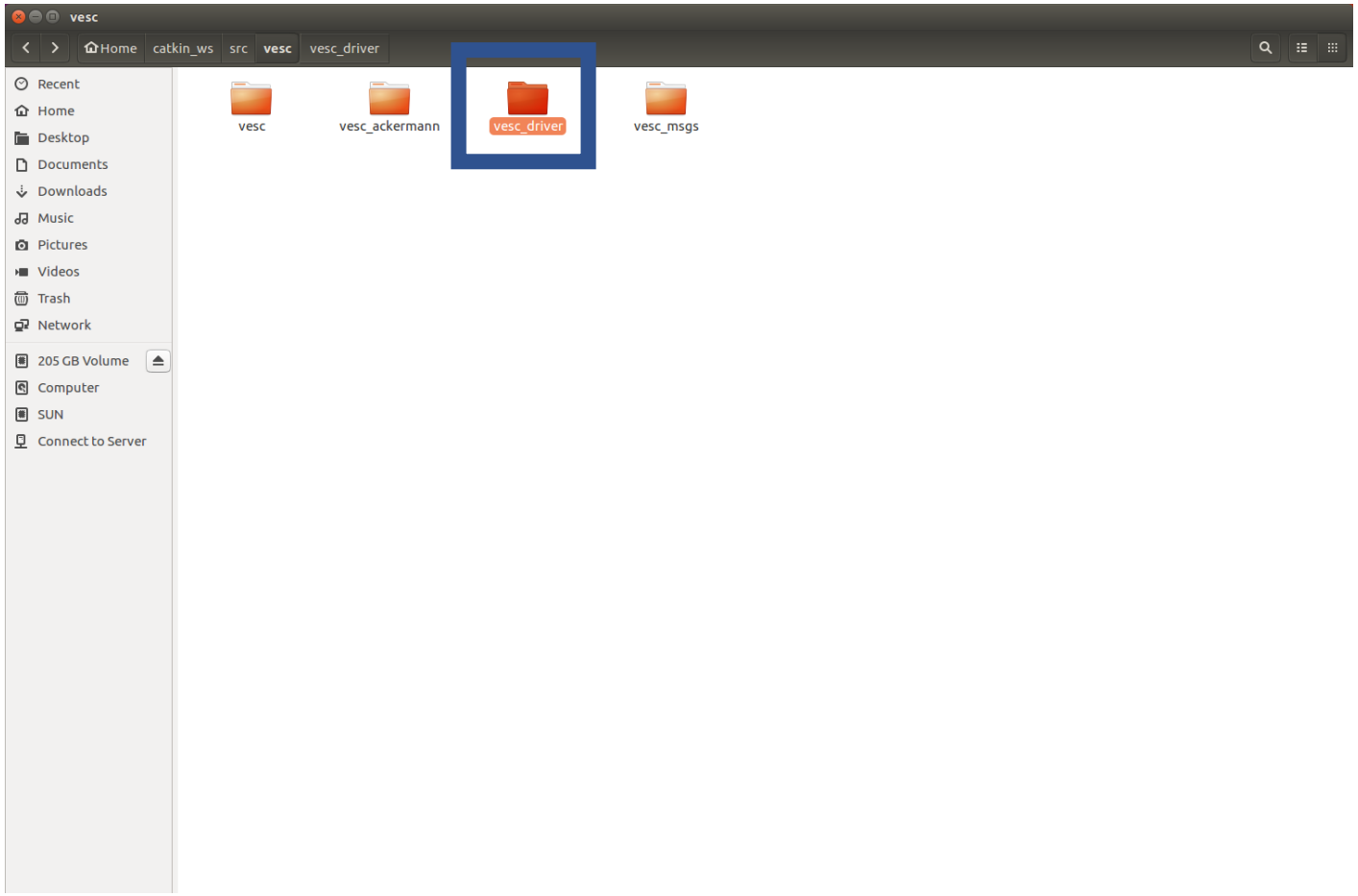


ROS

WeGo

VESC_pkg

- racecar_pkg

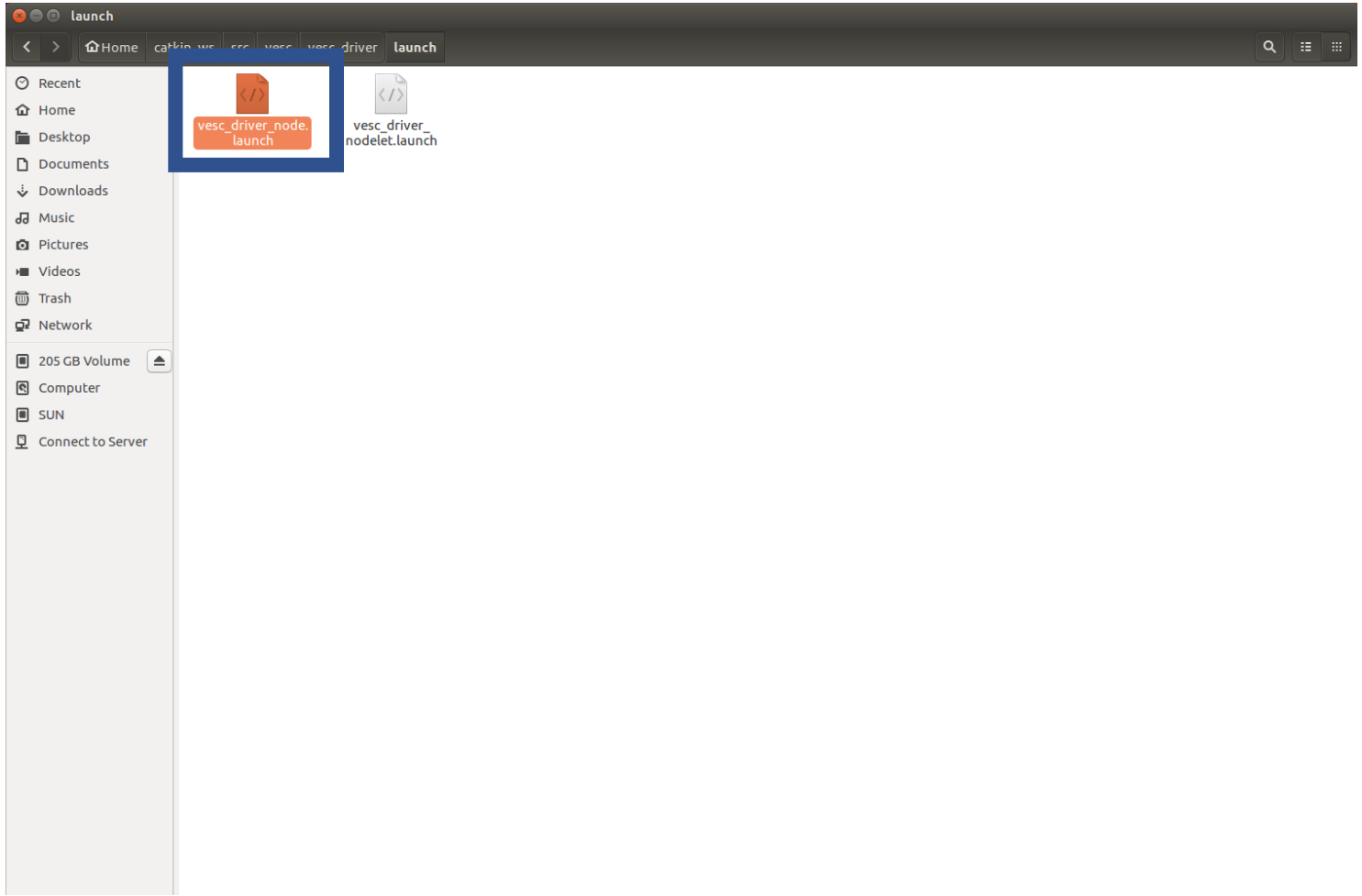


ROS

WeGo

VESC_pkg

- vesc_driver_pkg



VESC_pkg

- vesc_pkg_launch (Node name: vesc_driver)

vesc_driver_node.launch

home > js > catkin_ws > src > vesc > vesc_driver > launch > vesc_driver_node.launch

```xml
<!-- -*- mode: XML -*- -->
<launch>

  <arg name="node_name" default="vesc_driver_node" />

  <!-- Optionally launch in GDB, for debugging -->
  <arg name="debug" default="false" />
  <arg if="$(arg debug)" name="launch_prefix" value="xterm -e gdb --args" />
  <arg unless="$(arg debug)" name="launch_prefix" value="" />

  <!-- VESC driver parameters -->
  <arg name="port" default="/dev/ttyACM1" />

  <!-- VESC driver node -->
  <node pkg="vesc_driver" type="vesc_driver_node" name="$(arg node_name)"
        output="screen" launch-prefix="$(arg launch_prefix)" >
    <param name="port" value="$(arg port)" />
  </node>

</launch>
```

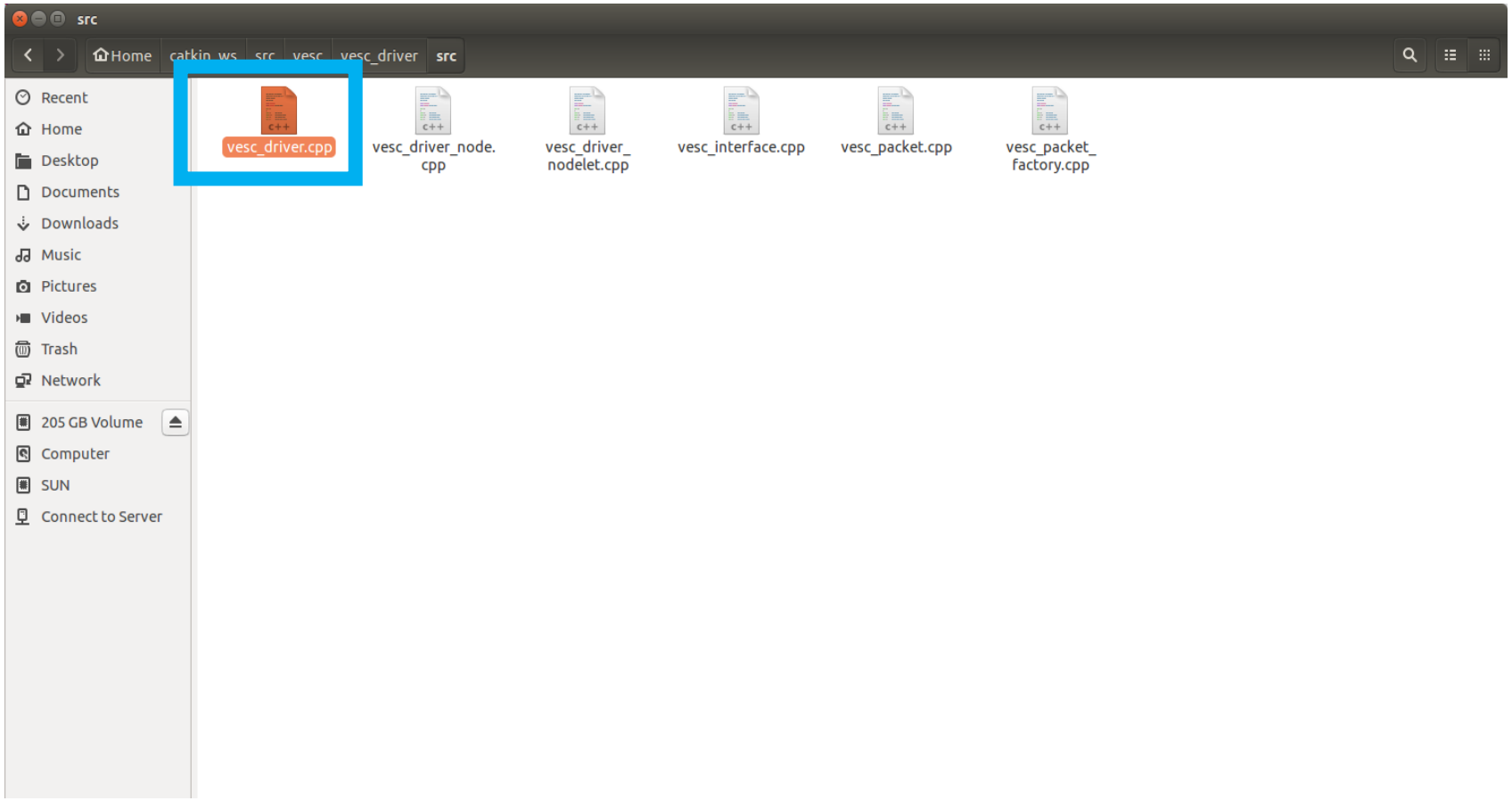
master* Python 2.7.12 64-bit 0 0 Ln 1, Col 1 Spaces: 2 UTF-8 LF XML

ROS

WeGo

VESC_pkg

- vesc_drive.cpp (모터컨트롤 드라이버)



ROS

WeGo

VESC_pkg

- vesc_node_name (이름 설정)

*vesc_driver_node.cpp (~/catkin_ws/src/vesc/vesc_driver/src) - gedit

Open

```
1 #include <ros/ros.h>
2
3 #include "vesc_driver/vesc_driver.h"
4
5 int main(int argc, char** argv)
6 {
7   ros::init(argc, argv, "vesc_driver_node");
8   ros::NodeHandle nh;
9   ros::NodeHandle private_nh("~");
10
11   vesc_driver::VescDriver vesc_driver(nh, private_nh);
12
13   ros::spin();
14
15   return 0;
16 }
17
```

VESC_pkg

- vesc_motor_commands

Open Save

```cpp
// -*- mode:c++; fill-column: 100; -*-

#include "vesc_driver/vesc_driver.h"

#include <cassert>
#include <cmath>
#include <sstream>

#include <boost/bind.hpp>
#include <vesc_msgs/VescStateStamped.h>

namespace vesc_driver
{

VescDriver::VescDriver(ros::NodeHandle nh,
                       ros::NodeHandle private_nh) :
  vesc_(std::string(),
        boost::bind(&VescDriver::vescPacketCallback, this, _1),
        boost::bind(&VescDriver::vescErrorCallback, this, _1)),
  duty_cycle_limit_(private_nh, "duty_cycle", -1.0, 1.0), current_limit_(private_nh, "current"),
  brake_limit_(private_nh, "brake"), speed_limit_(private_nh, "speed"),
  position_limit_(private_nh, "position"), servo_limit_(private_nh, "servo", 0.0, 1.0),
  driver_mode_(MODE_INITIALIZING), fw_version_major_(-1), fw_version_minor_(-1)
{
  // get vesc serial port address
  std::string port;
  if (!private_nh.getParam("port", port)) {
    ROS_FATAL("VESC communication port parameter required.");
    ros::shutdown();
    return;
  }

  // attempt to connect to the serial port
  try {
    vesc_.connect(port);
  }
  catch (SerialException e) {
    ROS_FATAL("Failed to connect to the VESC, %s.", e.what());
    ros::shutdown();
    return;
  }

  // create vesc state (telemetry) publisher
  state_pub_ = nh.advertise<vesc_msgs::VescStateStamped>("sensors/core", 10);

  // since vesc state does not include the servo position, publish the commanded
  // servo position as a "sensor"
  servo_sensor_pub_ = nh.advertise<std_msgs::Float64>("sensors/servo_position_command", 10);

  // subscribe to motor and servo command topics
  duty_cycle_sub_ = nh.subscribe("commands/motor/duty_cycle", 10,
                                 &VescDriver::dutyCycleCallback, this);
  current_sub_ = nh.subscribe("commands/motor/current", 10, &VescDriver::currentCallback, this);
  brake_sub_ = nh.subscribe("commands/motor/brake", 10, &VescDriver::brakeCallback, this);
  speed_sub_ = nh.subscribe("commands/motor/speed", 10, &VescDriver::speedCallback, this);
  position_sub_ = nh.subscribe("commands/motor/position", 10, &VescDriver::positionCallback, this);
  servo_sub_ = nh.subscribe("commands/servo/position", 10, &VescDriver::servoCallback, this);

```

C++ Tab Width: 8 Ln 2, Col 1 INS

ROS

WeGo

VESC_pkg

- vesc_drive.cpp (subscribe – brake, speed, position)속도 및 조향

vesc_driver.cpp

home > js > catkin_ws > src > vesc > vesc_driver > src > vesc_driver.cpp

```cpp
  // create vesc state (telemetry) publisher
  state_pub_ = nh.advertise<vesc_msgs::VescStateStamped>("sensors/core", 10);

  // since vesc state does not include the servo position, publish the commanded
  // servo position as a "sensor"
  servo_sensor_pub_ = nh.advertise<std_msgs::Float64>("sensors/servo_position_command", 10);

  // subscribe to motor and servo command topics
  duty_cycle_sub_ = nh.subscribe("commands/motor/duty_cycle", 10,
                                 &VescDriver::dutyCycleCallback, this);
  current_sub_ = nh.subscribe("commands/motor/current", 10, &VescDriver::currentCallback, this);
  brake_sub_ = nh.subscribe("commands/motor/brake", 10, &VescDriver::brakeCallback, this);
  speed_sub_ = nh.subscribe("commands/motor/speed", 10, &VescDriver::speedCallback, this);
  position_sub_ = nh.subscribe("commands/motor/position", 10, &VescDriver::positionCallback, this);
  servo_sub_ = nh.subscribe("commands/servo/position", 10, &VescDriver::servoCallback, this);

  // create a 50Hz timer, used for state machine & polling VESC telemetry
  timer_ = nh.createTimer(ros::Duration(1.0/50.0), &VescDriver::timerCallback, this);
}

/* TODO or TO-THINKABOUT LIST
   - what should we do on startup? send brake or zero command?
   - what to do if the vesc interface gives an error?
   - check version number against know compatable?
   - should we wait until we receive telemetry before sending commands?
   - should we track the last motor command
   - what to do if no motor command received recently?
   - what to do if no servo command received recently?
```

master* Python 2.7.12 64-bit ⊗ 0 ⚠ 0 Ln 1, Col 1 Spaces: 2 UTF-8 LF C++

ROS

WeGo